Numerical solutions of functions

One of the things that we often need to do in physics is to find the roots of functions. In some simple cases, this can be done analytically. For example, you can use:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{1}$$

to find the roots of a quadratic equation of the form:

$$ax^2 + bx + c = 0 \tag{2}$$

There are formulae, albeit more complicated ones, that work for values up to 4th order polynomicals. It has been proved, in the Abel-Ruffini Theorem, that 5th order polynomials cannot be solved with a catch-all formula. There are also, of course, functions which are not polynomials which can be even harder to solve by this kind of methodology.

In this week's assignment, we will program up one of the classic methods, something called the Newton-Raphson method. Before we get to it, though, I want to start off by giving some warnings about some of the problems with that method, and some discussion of the reasons for the existence of other methods, which are often slower to give a good answer, and which are definitely more complicated to program. I will also give you some discussion about how you can use the Newton-Raphson method more "safely".

A general numerical approach to finding roots

First, we want to consider a general algorithmic approach to finding the roots of an equation. The ideal approach will be simple to program, and will always find the root that we want to find. We need to remember that some functions have no roots, and we'd like to figure that out right away, and some functions have many roots, and we'd like to make sure that if we don't find all of them, we find that one or ones that we want.

Let's take a look at a few cases that can present special challenges:

1. Functions which don't have any roots. An example is:

$$f(x) = \frac{1}{x - c}. \tag{3}$$

   This function approaches $\pm\infty$ as it approachs c from the different sides of c, but it never actually takes the value 0.

2. Functions with roots close to one another. An example might be:

$$f(x) = x^2 - 0.0001. \tag{4}$$

   In this case there will be two roots, at ± 0.01. In a simple case like this, it's fairly straightforward to deal with this problem, but with a more complicated function that gave these two values, and for which it wasn't

clear there were two solutions, it might be hard to determine ahead of time that there were really two nearby solutions. This could be especially true if you had a function with solutions at 0.01, 0.011, and 500.

3. Functions with derivatives that behave in complicated ways. As we will see in a bit, the Newton-Raphson method makes use of the derivative of a function to help find the roots. If the derivative flips signs near the root, the Newton-Raphson method can be very difficult to use effectively.

There are some simple methods that can work and which can be programmed easily, but which converge slowly, and hence which we will not use. One is the bisection method. In this method, the algorithmic approach is as follows:

1. Find a value x_+ for which $f(x_+) > 0$ and a value x_- for which $f(x_-) < 0$.
2. Compute the value of $f(x_+ + x_-)$. Some small fraction of the time, the value of the function will be exactly zero, but otherwise, one will have either a positive or a negative value of f at that halfway point.
3. Ask if the precision on the answer is suitable for your purpose. If it is, then stop. If not, go back to step 2, using the new value you have found.

This method always works. It can run into trouble for cases where there are two close together roots within your search. Then it will find one of them. If there's a singularity, rather than a root, it will find the singularity's location to high precision. By seeing that the evaluations of the function are going toward infinity, rather than toward zero, as you converge, you should also be able to figure out that the system is converging to a singularity.

A faster, but less robust approach: the Newton-Raphson method

The Newton-Raphson method is a simple, straightforward method for solving equations which have smoothly varying derivatives. Mathematically, it comes form taking the Taylor series of a function around the root:

$$f(x+\delta) \approx f(x) + f'(x)\delta + \frac{f''(x)}{2}\delta^2 + ... \tag{5}$$

If we are already relatively close to the root, so that we can ignore all but the first two terms on the right hand side of the equation, we can say that:

$$\delta == \frac{f(x)}{f'(x)}. \tag{6}$$

We can then turn this into an algorithmic approach for finding the roots to equations:

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}. \tag{7}$$

We can iterate through the above equation and for "well-behaved" functions, we should find that we quickly converge to the right root.

There are now two more things to consider: first, what constitutes a well-behaved function, and second, what do we do in cases where we can't compute the function's derivative.

The meaning of "well-behaved"

The definition here will be a bit circular. A function is well-behaved with respect to the Newton-Raphson method if it converges to giving the right answer. This isn't particularly helpful, but what can be useful is to consider some cases which aren't well behaved. Consider, for example, a cubic function, which has a region of zero derivative somewhere close to, but not at, the solution. In that case, you could get the derivative taking the wrong sign, even.

The solution for how to deal with this case is to have a good initial guess at the value of the root. You can ordinarily do this by starting by graphing the function, and eyeballing where the root is, and then using that as a starting value.

When we can't compute the derivative

Sometimes we will have functions which do not have derivatives which can be computed analytically. This will not be the case for any "simple" functions, but it may be the case for some more complicated special functions that may be interesting in physics. In this case, we can go back to the definition of a derivative and compute the derivative numerically:

$$f'(x)dx = f(x + dx) - f(x). \tag{8}$$

This has the advantage of being something that works for any function that can be computed. It has the disadvantages of requiring more computer time per step, and also of being less precise that computing the derivative analytically.

Additional reading If you wish to get a better feel for how to work with more complicated methods, please read chapter 9 of *Numerical Recipes*, which covers a wide range of approaches. A fair bit of these notes were based on the discussions in *Numerical Recipes*.